Assignment 4

CS-GY 6033 INET Fall 2024

Due date:
11:55pm on November 16th, 2024 **on Gradescope**.

Instructions:

Dynamic Programming

For each of the dynamic programming problems below, you must:

- Define the table you are using, and define each entry
- Describe how to initialise your table
- Describe the relationship between the entries in your table
- Describe which entry in the table stores your final result
- Provide the pseudocode that shows how to fill up your table
- Justify the runtime of your algorithm

Question 1

10 points

In this question, you will provide a solution to a variation of the rod cutting problem from class. Let $p[1, 2, \ldots, n]$ be an array representing the price of each piece cut from the rod: $p[i]$ is the price of a piece of size i. Given a rod of length n, your job is to write a dynamic programming solution that find the maximum price achievable when cutting the rod into integer-sized pieces. The new condition is that a particular piece size can **only be cut once!**. For example, you are not allowed to have cut two pieces of the same size.

Question 2

(a) 12 points A game player is sitting at position $(1, 1, 1)$ on a 3D grid. The player takes one step at a time, where a step increases *exactly one* coordinate by 1. In other words, the player can either increase the x coordinate by 1, or the y coordinate by 1, or the z coordinate by 1. The player makes a sequence of steps and eventually arrives at position (m, n, p). The input $A[1 \ldots m, 1 \ldots n, 1 \ldots p]$ is a 3D array, where $A[i, j, k]$ represents the number of assasins at position (i, j, k). The more assasins, the most likely you are to be killed! Note that some positions are completely safe (with no assasins), in which case $A[i, j, k] = 0$. Suppose that each assasin shoots with a 10% accuracy, meaning that that survival rate on a spot with a single assasin is 0.9, but your survival rate on a spot with k assasins is 0.9^k. Your job is to design a DP solution that determines your highest chance of survival, in travelling from $(1, 1, 1)$ to (m, n, p). Call your procedure MaxSurvival($A[1..m, 1..n, 1..p)$ which returns the maximum survival rate. In your solution, define the table $T[i, j, k]$ which represents the maximum survival rate from $(1, 1, 1)$ to (i, j, k).

(b) 8 points Consider the above problem. Write the pseudo-code for the procedure that outputs the **safest route** from $(1, 1, 1)$ to (m, n, p). It is acceptable if the output is printed in reverse order. For example, if $(m, n, p) = (5, 5, 5)$, the output route could be $(5, 5, 5), (5, 4, 5), (5, 3, 5), (5, 3, 4)$..., etc.

Question 3

(a) 12 points Given a set of n boxes, where each box has a base (with 2 dimensions) and height, the problem is to determine the height of the largest tower that can be built by placing the boxes one on top of each other. A box can be placed on top of another box, as long as both dimensions of the lower base are greater than or equal to the base dimensions of the added box. For example, a box with base 2×3 can be placed on a box of base size 3×4, but NOT on top of a box with base size 1×5. Note that the boxes cannot be rotated, because base is distinct from the height. The input is given in the arrays: $L[1, 2, \ldots, n]$, $W[1, 2, \ldots, n]$, $H[1, 2, \ldots, n]$ where the base dimensions of box i are $L[i] \times W[i]$ and the heigh of box i is $H[i]$. Design a dynamic programming solution that solves the problem of finding the height of the largest tower.

(b) 8 points Write the pseudo-code for the procedure that outputs the indices of the boxes that are used in the highest tower. The output must be in the order from largest to smallest.

Question 4

15 points An island has only three species living on it: the Komodo dragon, the wild boar, and the Coywolf. These animals roam the forests of the island, and periodically encounter each other. If a dragon encounters a boar, the boar kills the dragon. If a boar encounters a coywolf, the wolf kills the boar. If a coywolf encounters a dragon, the dragon kills the wolf. Suppose initially on the island there are n dragons, m boars, and p coywolves. As they roam the island, any two individuals meet with equal probability. The chance that a dragon and a boar meets depends on how many dragons and boars there are! You may use the following (if your probability is not up to snuff...): the chance that a boar meets a dragon is:

$$\frac{nm}{\binom{n+m+p}{2}}$$

Eventually, there will be only **one** species left on the island. Your job is to determine the probability that each species is the last surviving species. For example, if the island starts with one animal of each species, there is a $1/3$ chance that the dragon and wolf meet first, in which case the dragon survives. Next, the dragon and the boar meet, and the boar survives. Therefore the surviving animal is the boar, with probability $1/3$. Therefore, for the input $m = 1, n = 1, p = 1$ the return value is $[1/3, 1/3, 1/3]$, meaning there is an equal chance that each species is the last surviving species. Your return value must be of the form $[a, b, c]$ where a is the chance that the dragon is the last species, b is the chance that the boar is the last species, and c is the chance that the wolf is the last species.

Question 5

15 points Consider a sequence of n days, where a specific commodity has price on each of those days. For example, $c[1, \ldots, n]$ represents the price of a commodity over a period of n days, $c[i]$ being the price on day i. You can chose to buy the commodity on any day, and sell it on any future day. You may also chose to buy again, and sell again, before the n days are up. However, at any one time, you can own at most one share of the commodity. Therefore, you can't make a second purchase, until you have sold your previous purchase. Suppose the prices over 10 days are $[3, 2, 5, 4, 7, 1, 5, 6, 4, 9]$. You may decide to buy on day one, and sell on day five, then buy on day six and sell on day ten. Your total profit is then $(7 - 3) + (9 - 1) = 12$, and you carried out exactly two transaction.

Your Job: Design a DP solution that finds the maximum profit obtainable for the case where you may carry out at most k transactions. The input to the problem is the commodity prices: $c[1, 2, \ldots, n]$ and the transaction number k.